extending of the call. As a result of the CMA_SLP instantiation, the NOS NT sends the call identification data and SLP address list (ELP, CLP, and LLP) to the new CMA SLP. Then, the advanced 18C SLP terminates and hands off this call to the CMA SLP.--.

Please replace the paragraph beginning on page 63, line 2, with the following:

--A 1-800 collect call ("18CC") service with a collect call option is now described in greater detail with respect to Figure 13(a). This 18CC scenario describes the ability to provide a 1-800 Collect service with options such as collect call and calling card options. To provide this functionality, this scenario implements an 18CC SLP which instantiates an LIDB Lookup SLP or SIBB ("LIDB_SLP") to verify that the called line is billable, and implements a validate direct dialed digits SLP or SIBB ("DDD_SLP") to verify that the DDD entered by the caller is valid. It is assumed that all database and voice files used in this scenario have been built using the NGIN Service Creation Environment.--.

Please replace the paragraph beginning on page 70, line 13, with the following:

--In the manner as described with respect to Figure 10(c), the next step 810 instantiates a BOC Card validation SLP or SIBB ("BOX_CC_SLP") which requests the validation of the BOC Card number entered by the caller. Once instantiated, the BOC CC SLP formats the query data to the appropriate format and forwards the query to the gateway to the BOC Card database. The BOC Calling Card query is executed and the result is returned to the 18CC SLP. For this scenario, it is assumed that the entered BOC Card number is valid.--.

REMARKS

The above amendments to specification are clerical in nature and do not add new matter. Therefore, Applicants respectfully request entry of the amendments.

Respectfully submitted,

Date: 7/3/2003

Brian C. Oakes
Registration No. 41,467

WorldCom, Inc.
1133 19th Street, NW
Washington, DC 20036
Phone: 202 736-6371
Fax: 202-736-6382

VERSION WITH MARKINGS TO SHOW CHANGES MADE

Please amend the paragraph beginning on page 1, line 8, as follows:

This application is a Continuation of U.S. Patent Application No. 09/420,669 filed October 19, 1999, now U.S. Patent 6,594,355, entitled "Method and Apparatus for Providing Real Time Execution of Specific Communications Services in an Intelligent Network" which is a Continuation-In-Part of commonly assigned, co-pending U.S. Patent Application No. 09/128,937, filed August 5, 1998, now U.S. Patent 6,418,461, [(MCI D# RIC-97-110)] entitled [] "Intelligent Call Platform for an Intelligent Network Distributed Architecture"[] which claims the benefit of U.S. Provisional Application Serial No. 60/061,173, filed October 6, 1997, both of which are incorporated herein in their entirety by reference thereto. This application additionally claims the benefit of U.S. Provisional Application Serial No. 60/104,890, filed October 20, 1998, the whole contents and disclosure of which is incorporated by reference as if fully set forth herein.

Please amend the paragraph beginning on page 8, line 25, as follows:

The IN and AIN architectures attempt to predefine a standard set of functions to support all foreseeable services. These standard functions are all hard-coded into various state machines in the switch. Unfortunately, any new functions, which are likely to arise in conjunction with new technologies or unforeseen service needs, cannot be implemented without an extensive overhaul and testing of the network software across many vendor platforms. Furthermore, if a new function requires changes to standardized call models, protocols, or interfaces, the implementation of the service utilizing that function may be delayed until the changes are ratified by an industry standards group. But even as draft standards have attempted to broaden the set of IN and AIN supported functions, equipment suppliers have refused to endorse these draft standards due to the staggering increase in code complexity. A detailed flow chart describing the process for generic service creation according to the prior art may be found in above-mentioned, commonly-owned, co-pending U.S. Patent Application No. 09/128,937, filed August 5, 1998, now U.S. Patent 6,418,461, [_____ (MCI D# RIC-97-110)] entitled []"Intelligent Call Platform for an Intelligent Network Distributed Architecture"[], the contents and disclosure of which is incorporated by reference as if fully set forth herein.

Please amend the paragraph beginning on page 10, line 19, as follows:

In another attempt to solve these problems, as disclosed in pending U.S. Patent Application Serial No. 08/580,712, filed August 30, 1999, now U.S. Patent 6,041,109, entitled

"Telecommunications System Having Separate Switch Intelligence and Switch Fabric", a Separate Switch Intelligence and Switch Fabric ("SSI/SF") architecture, which is referred to generally as 150 (Figure 1), logically separates the SSP 70 from the Switching System 44. Now referring back to Figure 1, the switch intelligence 152 contains the call processing functions 24 and facility processing functions 26 that are encoded in discrete state tables with corresponding hard-coded state machine engines, which is symbolized by circles 154 and 156. The interface between the switch fabric functions 158 and switch intelligence functions 152 may be extended through a communications network such that the switch fabric 158 and switch intelligence 152 may not necessarily be physically located together, by executed within the same processor, or even have a one-to-one correspondence. In turn, the switch intelligence 152 provides a consistent interface of simple non-service-specific, non-manufacturer-specific functions common to all switches.

Please amend the paragraph beginning on page 12, line 15, as follows:

The present invention is directed to a methodology for performing services in response to service requests, e.g., 1-800 telephone calls, received at a switch associated with a node of an intelligent communications network ([ ]"IN" [ ]). Particularly, the intelligent network includes a plurality of service nodes, each node providing an execution environment that may provide all of the call processing functionality necessary to handle a call at the instance it is received at the switch or resource complex physically associated with that particular service node.

Please amend the paragraph beginning on page 16, line 9, as follows:

The present invention is one component of a comprehensive intelligent network alternately referred to herein as the an Intelligent Distributed Network Architecture ("IDNA") or the Next Generation Intelligent Network ([ ]"NGIN"[ ]). As described herein, the NGIN architecture is designed to perform intelligent call processing services for any type of call received at a resource complex or switching platform, e.g., switch, router, IP termination address, etc. The IDNA/NGIN preferably comprises a plurality of distributed service nodes with each node providing an execution environment providing call processing functionality necessary to handle a call at the instance it is received at the switch or resource complex physically associated with that particular service node. NGIN is of a highly scalable architecture and engineered to ensure that executable service objects, embodied as independent Service Logic Programs ([ ]"SLP"[ ]), and associated data for performing event services, e.g., 1-800 telephone call, send fax, etc., may be deployed to and maintained at the service nodes in a cost-effective manner. By employing CORBA-compliant Object Request Broker technology, the intelligent network supports location and platform-independent call processing

service execution independent of and transparent to the event switching platform or resource complex in which an event or call is received, and, enables high-level logic programs to be run virtually anywhere in the network independent of the service execution platform. Furthermore, the system provides location-independent communications among these distributed processes.

Please amend the paragraph beginning on page 17, line 30, as follows:

Each IDNA Node 204 contains an Intelligent Call Processor ("ICP") 172 and a Resource Complex 180 (Figure 1). Figure 3 illustrates an IDNA Node 204 having a Resource Complex A ("RCA") 206 and a Resource Complex B ("RCB") 208. The ICP can be linked to Adjunct Processors 210, which provide existing support functions, such as provisioning, billing and restoration, however, these functions may be absorbed by functionality provided by a Network Management System ("NMS") 212. In the preferred embodiment, however, these support functions may be provided by a centralized Service Administration ([ ]"SA"[ ]) system 500 having Data Management ([ ]"DM"[ ]) component 400 as will be described herein with respect to Figure 4. As further shown in Figure 3, the ICP 172 can be also linked to other ICP's 172, other networks (not shown), or other devices (not shown) through a direct link 214 having signaling 216 and bearer links 218. A direct link prevents latency between the connected devices and allows the devices to communicate in their own language. The ICP 172 is the "brain" of the IDNA Node 204 and is preferably a general purpose computer, which may range from a single processor with a single memory storage device to a large scale computer network depending on the processing requirements of the IDNA Node 204. Preferably, the general purpose computer will have redundant processing, memory storage and connections.

Please amend the paragraph beginning on page 23, line 7, as follows:

As further shown in Figure 3, the Managed Object Creation Environment ("MOCE") 228 includes the sub-components to create services that run in the IDNA network 200. A Service Independent Building Block and API representations that a service designer uses to create new services are imbedded within the MOCE'S primary sub-component, a Graphical User Interface ("GUI"). The MOCE 228 is a unified collection of tools hosted on a single user environment or platform, alternately referred to as a Service Creation ([ ]"SC"[ ]) environment. It represents the collection of operations that are required throughout the process of service creation, such as service documentation, managed object definition, interface definition, protocol definition and data input definition, which are encapsulated in managed objects, and service testing. The network owner only has to develop a service once using the MOCE 228, because managed objects can be applied to all the

nodes on his network. This is in contrast to the network owner having each of the various switch manufacturers develop their version of the service, which means that the service must be developed multiple times.

Please amend the paragraph beginning on page 24, line 5, as follows:

In accordance with the preferred embodiment of the invention, as shown in Figure 4, the IDNA/NGIN system includes a centralized Service Administration ([ ]"SA"[ ]) component 500 that provides both a storage (Repository) 230 functionality and the generic network management (NMS) 212 functionality of the IDNA system 170 together with added capabilities as described in commonly-owned, co-pending U.S. Patent Application No. 09/412,590, filed October 20, 1999, [_____ (D#11337, COS-98-021)] entitled "Method And Apparatus For Deploying Service Modules Among Service Nodes Distributed In An Intelligent Network" [METHOD AND APPARATUS FOR DEPLOYING SERVICE MODULES AMONG SERVICE NODES DISTRIBUTED IN AN INTELLIGENT NETWORK], the contents and disclosure of which is incorporated by reference as if fully set forth herein. Generally, the SA component 500 as shown in Figure 4 supports off-line storage, naming, distribution, activation and removal of all services and data for the IDNA/NGIN system and, additionally provides a data management ([ ]"DM"[ ]) function enabling the run-time storage, replication, synchronization, and availability of data used by the service objects in the IDNA service nodes.

Please amend the paragraph beginning on page 28, line 20, as follows:

It should be understood that, collectively, the LOS and WANOS functionality may be represented as a Network Operating System or [ ]"NOS"[ ], as shown in Figure 6, that functions to provide platform independent and location independent connectivity between the IDNA/NGIN system components. That is, NOS comprises a set of network-wide services that provides process interfaces and communications among the other IDNA/NGIN functional components and sub-components. Among the services provided by NOS are object connectivity, logical name translation, inter-process communications, and local and system-wide resource management ([ ]"RM"[ ]). For instance, as shown in Figure 3, the NOS component 700 provides the local (NODE RM) and system-wide resource management (SYS RM) function, as described in commonly-owned, co-pending U.S. Patent Application No. 09/420,654, filed October 19, 1999, now U.S. Patent 6,425,005, [_____ (D#11357, COS-98-029)] entitled "Method And Apparatus For Managing Local Resources In Service Nodes Of An Intelligent Network," [METHOD AND APPARATUS FOR MANAGING LOCAL RESOURCES IN SERVICE NODES OF AN INTELLIGENT NETWORK,] the contents and

disclosure of which is incorporated by reference as if fully set forth herein. Particularly, the NOS component encapsulates the location of any service from the processes that need services and data, so that a process only needs to make a call to a single logical name. The NOS component then determines which instance of a service to use, and provides connectivity to that instance. The NOS 700 enables, in part, both the widely distributed nature of IDNA/NGIN, and the platform-independence of IDNA/NGIN. For example, the aforementioned logic programs use the NOS component 700 to call other logic programs, and can therefore call and invoke other logic programs that run on different SLEEs either in the same service node or a remote service node. Particularly, through the SA 500, a service node may be specified to perform only certain services. When a call that arrives at a switch having an associated service node 204 for which the needed service may not be performed, e.g., joining a conference bridge, IDNA may need to route the call to another node configured to provide such service. Preferably, IDNA, via the NOS component 700, will call the needed service at another remote service node, perform the call processing, and provide a service response to the switch at the original node.

Please amend the paragraph beginning at page 38, line 26, as follows:

As shown in Figure 9, the SLEE 450 is designed to execute at least five types of logic programs implemented in performing call processing services and other supporting services: 1) Feature Discriminator logic programs ([ ]"FD"[ ]) 510, which are functional sub-components (objects) of the service control class/service discriminator class 296 (Figure 8) that first receive a service request from the switching platform, determine which service to perform on a call based on some available criteria, for example, the dialed number of the call, and, then calls on another appropriate Service Logic Program to process the call; 2) the Service Logic Program ([ ]"SLP"[ ]) objects 520, which are functional sub-components of the service control class 252 (Figure 8) that perform service processing for a received service request or event; 3) Line Logic Program ([ ]"LLP"[ ]) objects 530, which are functional sub-components of the call control class 250 (Figure 8) that maintain the current state of a network access line; 4) Event Logic Program ([ ]"ELP"[ ]) objects 540, which are functional sub-components of the service control/session manager class 260 (Figure 8) to which all other logic programs write events; and 5) Call Logic Program ([ ]"CLP"[ ]) objects 545 which are functional sub-components of the service control/connection manager class 302 (Figure 8) that maintains the state of an entire call by providing a connection point for all other logic programs that are involved in the processing of a call. Each of these logic programs are embodied as a software [ ]"objects"[ ], preferably written in Java™[ ] programming language, that may either be temporarily instantiated or persistent, as will be described. The IDNA/NGIN service control

architecture is designed such that these objects are written only once in MOCE/SCE, and may be deployed to a SLEEs on any type of computer and on any type of operating system anywhere in the network.

Please amend the paragraph beginning at page 44, line 13, as follows:

As shown in Figure 9, other processes that execute within the SLEE 450 for support and operational functions include: a Service Manager ([ ]"SM"[ ]) object 554, responsible for loading, activating, de-activating and removing services that run in the SLEE and, further monitoring all other services running within its SLEE, and reporting status and utilization data to NOS; a NOS client process 558 which is a NOS class library that is used for interfacing with NOS services and is used by all services running within that SLEE to call on NOS services, i.e., is the gateway to NOS; a thread manager (TM) 557, which provides functionality needed for NGIN services to execute concurrently without tying up all the SLEE resources; and, a Data Management API 410 used to interface with the local cache and cache manager components of DM 400 through the intermediary of the DMAPI 410. As an example, a 1-800-number service having a SIBB that has collected 1-800- number digits, for example, may need to interact with the data management component to query a database to perform a number translation. This is accomplished through the DM API 410 which will perform the translation look-up and provide the data back to the service. As described herein, the database may have been previously loaded to the local cache 415 or, the data is accessed from the local DBOR through the DM server 425.

Please amend the paragraph beginning at page 45, line 6, as follows:

Still other service instances loaded in the SLEE as shown in Figure 9 include a service agent instance 559 and a thread manager instance 557 associated therewith. Commonly-owned, co-pending U.S. Patent Application No. 09/420,654, filed October 19, 1999, now U.S. Patent 6,425,005, [_____ (D#11357, COS-98-029)] entitled "Method And Apparatus For Managing Local Resources In Service Nodes Of An Intelligent Network," [METHOD AND APPARATUS FOR MANAGING LOCAL RESOURCES IN SERVICE NODES OF AN INTELLIGENT NETWORK,] the contents and disclosure of which is incorporated by reference as if fully set forth herein describes, in greater detail the service activation process. As described in co-pending U.S. Patent Application No. 09/420,654, [_____ (D#11357, COS-98-029),] as part of the service activation, whenever a request is made for a service, e.g., in response to a call event, that requested service's service agent instance 559 will get the location of the call from NOS, via the NOS agent 558, and will query its thread manager instance 557 to determine if there is another thread instance that could process that call. For

example, a particular type of service may be allocated a predetermined number of instances, i.e., threads that may be invoked. The thread manager instance 557 will return a thread object (not shown) and clone an instance of that requested service (SLP) if the resources are available, or else will reject the call. If the new instance of the SLP is created, it's code is executed inside the thread. It should be understood that during this instantiation, a unique transaction identifier or session i.d. is created to determine which instance of the thread corresponds to the received call. Moreover, call context data is managed by the thread. Besides assigning execution threads, the service agent collaborates with the thread manager to monitor these threads and determine overall system loads.

Please amend the paragraph beginning on page 46, line 11, as follows:

Figures 10(a) - 10(i) describe the basic functional blocks implemented by the NGIN in the performance of services, e.g., calls, received at a network switch of the resource complex. These functional building blocks are generic in the sense that they may be implemented regardless of the type of service being performed and, particularly, they are described herein in the context of a 1-800/888 toll free call ([ ]"18C"[ ]), 1-800 collect call, etc. It is understood that with various modifications as described, the functional building blocks may be implemented in many event service scenarios.

Please amend the paragraph beginning on page 46, line 22, as follows:

First, as shown at step 601, Figure 10(a), it is assumed that a received call arrives at a Next Generation Switch ([ ]"NGS"[ ]) associated with a service node as described in greater detail in commonly-owned, co-pending U.S. Patent Application No. 08/580,712, entitled [ ]"A Telecommunications System Having Separate Switch Intelligence and Switch Fabric"[ ] the entire contents and disclosure of which is incorporated by reference as if fully set forth herein. As described in co-pending U.S. Patent Application No. 08/580,712, when the NGS switch 75 receives a call, a bearer control component provides the call control component with the access line on which the call was received, as well as the ANI, dialed number, and other data needed for call processing. Call control maintains a state model for the call, as executed in accordance with its programmed logic. Additionally included in the state model are triggers for instantiating an ELP 540 and sending a service request to a feature discriminator service (FD) 510 as shown in Figure 16 in the manner as will be described.

Figure 10(a) is a sequence diagram describing the steps for performing feature discrimination on an incoming call. As shown at step 610, a logical name for the FD is sent from an NGS/NOS agent object to the NOS Name Translation (NT) function. Preferably, this Initial Address Message

message includes both the name and the data (envelope and letter) with additional data such as the called 800#, ANI, Line ID, Network Call ID, Originating Switch Trunk. An ELP address is also sent along in this information. As indicated at step 612, a Name Translation is performed by NT to determine the feature discriminator name. It sends that name to DM to get the actual SLP name, i.e., FD.SLP). In this scenario, it is assumed that there is a feature discriminator in each SLEE that is always running (i.e., a persistent SLP). Then, as indicated at step 614, Data Management communicates the actual name of the FD SLP with its stored locations to the Name Translator (NT) which, in turn, sends the name to the NOS LRM function at step 616 to determine where the FD SLP is instantiated. It is understood that if a FD is not instantiated, NOS will instantiate one. The LRM picks a SLEE and returns the address of the SLEE to NT SLEE Address) as indicated at step 618. Then, at step 620, the NOS NT then sends the message (that came from NGS) to the Feature Discriminator SLP containing all the call origination information that came in. As part of this functionality, as indicated at step 625, the FD SLP then performs an FD database ([▒]"DB"[▒]) lookup so that it may make a logical decision.

Please amend the paragraph beginning on page 49, line 8, as follows:

Particularly, in the context of the 18C service request, an FD SLP uses its feature discrimination table to identify which SLP is to handle the received service request. For example, if the received message is a 18C service request, it is to be handled by the 18C SLP. Table 3 below is an example abbreviated FD table having entries including pointers to various [▒]"toll-free", e.g., 1-800, call services.

Please amend the table beginning on page 49, line 17, as follows:

Entry Port Table

[▒]"001001" SLP pointer 'Vnet'
[▒]"001002" Table pointer to FGD table

FGD table

1800* table pointer 800 table
1888* table pointer 800 table
1900* table pointer 900 table
1* SLP pointer 'Local number'

800 table
1800collectSLP pointer to '1-800-C'
18008888000SLP pointer 'Op Service'
1800* SLP pointer '800 service'

1888* SLP pointer '800 service'

Please amend the paragraph beginning at page 49, line 35, as follows:

where FGD is the feature group discriminator. Particularly, based on where the call originated in the network (switchboard) and the type of call received (e.g., 1-800), the FD will determine an appropriate SLP logical name. For instance, the identification [ ]"001002" indicates receipt of a call requiring a look-up in the FGD table (pointer to FGD table). The FGD table in turn, maintains pointers to other tables depending upon the called number, e.g., 800* where '*' is a delimeter. From this 800 table, for example, the FD obtains a pointer to the requested SLP logical name as indicated at step 649. Subsequently, this SLP is invoked and the service request is handed off to NOS which instantiates a CLP 545, LLPO 530 and the SLP 520 objects according to the 18C service requested.

Please amend the paragraph beginning on page 60, line 5, as follows:

Building on the advanced 18C scenario, another SLP may be executed to play a message to the caller first before extending the call to its termination. Figure 12(a) illustrates this advanced 18C service scenario implementing customized message announcement and call extension features. First, the advanced 18C SLP described with respect to Figure 11(a) is instantiated for the 800 number translation. Particularly, as indicated at step 732, this involves: receiving the intelligent request at the switch, performing feature discrimination, and, performing the advanced 18C SLP and LLP (and CLP) object instantiations. Assuming the instantiated advanced 18C SLP determines no features associated with the originating line, then, a lookup is performed to determine the correct routing. As part of this routing query, a customer profile lookup is first done, as indicated at step 733 followed by a day and percent allocation query, as indicated at step 734. As a result of the day and percent allocation query, DM returns routing instructions for a call extension and the name of the new Customized Message Announcement SLP ([ ]"CMA SLP"[ ]) for handling the remainder of the call to the advanced 18C SLP. Then, as indicated at step 735, the terminating node location is determined, and, any call context data may be written to the ELP at this point for placement in the call context DM.

Please amend the paragraph beginning on page 60, line 30, as follows:

Then, as indicated at step 736, the new Customized Message Announcement SLP ([ ]"CMA SLP"[ ]) is instantiated. This CMA SLP invokes SIBBs to direct the playing of the voice file and the extending of the call. As a result of the CMA_SLP instantiation, the NOS NT sends the call

identification data and SLP address list (ELP, CLP, and LLP) to the new CMA SLP. Then, the advanced 18C SLP terminates and hands off this call to the CMA SLP.

Please amend the paragraph beginning on page 63, line 2, as follows:

A 1-800 collect call ("18CC") service with a collect call option is now described in greater detail with respect to Figure 13(a). This 18CC scenario describes the ability to provide a 1-800 Collect service with options such as collect call and calling card options. To provide this functionality, this scenario implements an 18CC SLP which instantiates an LIDB Lookup SLP or SIBB ([▒]"LIDB_SLP"[▒]) to verify that the called line is billable, and implements a validate direct dialed digits SLP or SIBB ([▒]"DDD_SLP"[▒]) to verify that the DDD entered by the caller is valid. It is assumed that all database and voice files used in this scenario have been built using the NGIN Service Creation Environment.

Please amend the paragraph beginning on page 70, line 13, as follows:

In the manner as described with respect to Figure 10(c), the next step 810 instantiates a BOC Card validation SLP or SIBB ([▒]"BOX_CC_SLP"[▒]) which requests the validation of the BOC Card number entered by the caller. Once instantiated, the BOC CC SLP formats the query data to the appropriate format and forwards the query to the gateway to the BOC Card database. The BOC Calling Card query is executed and the result is returned to the 18CC SLP. For this scenario, it is assumed that the entered BOC Card number is valid.